

# Introdução ao L<sup>A</sup>T<sub>E</sub>X

Seminário L<sup>A</sup>T<sub>E</sub>X - o Livro

Geraldo Xexéo<sup>1,2</sup>

<sup>1</sup>Departamento de Ciências da Computação

<sup>2</sup>Programa de Engenharia de Sistemas e Computação

Março 2020



# Sumário

<b>1</b>	<b>Introdução ao Processamento de Texto</b>	<b>7</b>
1.1	Tipos de Sistema de Processamento de Texto . . . . .	7
1.1.1	Editores de Texto . . . . .	8
1.1.2	IDEs . . . . .	9
1.1.3	Sistemas de Composição ( <i>Typesetting</i> ) . . . . .	11
1.1.4	Processadores de Texto . . . . .	12
1.1.5	Sistemas de Autoria . . . . .	13
1.1.6	Sistemas de Publicação . . . . .	14
1.1.7	Sistemas Colaborativos de Edição . . . . .	15
1.2	Tipos de Linguagens para Arquivos de Texto . . . . .	16
1.2.1	Linguagens de Marcação . . . . .	17
1.2.2	Linguagens de Impressão . . . . .	18
1.3	O que é o $\text{\LaTeX}$ . . . . .	18
1.4	Por que $\text{\LaTeX}$ ? . . . . .	19
1.4.1	O Design Lógico de Documentos . . . . .	20
1.5	Sistemas Mais Usados na Computação . . . . .	20
1.6	Mitos e Fatos de Word vs $\text{\LaTeX}$ . . . . .	21
1.6.1	$\text{\LaTeX}$ é mais produtivo . . . . .	21
1.6.2	A qualidade de saída do $\text{\LaTeX}$ é muito melhor . . . . .	21
1.6.3	Word não trabalha bem com fórmulas matemáticas . . . . .	21
1.6.4	Você perde muito tempo com besteira no $\text{\LaTeX}$ . . . . .	21
1.6.5	Não consigo colocar a imagem onde quero no Word . . . . .	21
1.6.6	Em $\text{\LaTeX}$ gerencio melhor as bibliografias . . . . .	22
1.6.7	Em $\text{\LaTeX}$ consigo controlar versões . . . . .	22
1.6.8	Word é mais fácil de aprender . . . . .	22
1.6.9	Com o Word, basta ele . . . . .	22
1.6.10	Word tem problemas com arquivos grandes . . . . .	22
1.7	Recomendações . . . . .	22

<b>2</b>	<b>O Mundo <math>\text{\LaTeX}</math></b>	<b>25</b>
2.1	Invenção do $\text{\LaTeX}$ . . . . .	25
2.2	Situação Atual do $\text{\LaTeX}$ . . . . .	26
2.3	Como funciona o $\text{\LaTeX}$ . . . . .	28
2.4	Recomendações de uso . . . . .	29
<b>3</b>	<b><math>\text{\LaTeX}</math> Básico</b>	<b>31</b>
3.1	$\text{\LaTeX}$ Muito Básico . . . . .	31
3.1.1	Documento Mínimo . . . . .	31
3.2	Estrutura de um documento . . . . .	32
3.3	Informação de Capa . . . . .	33
3.4	Comandos mais comuns . . . . .	35
3.4.1	Fazendo referência a outra parte do documento . . . . .	35
3.5	Usando pacotes . . . . .	37
3.5.1	O Que São Pacotes . . . . .	37
3.5.2	Babel . . . . .	38
3.5.3	inputencode . . . . .	38
3.5.4	fontencode . . . . .	38
3.6	Ambientes . . . . .	39
3.6.1	Ambiente Mais Usados . . . . .	39
3.6.2	Equações . . . . .	39
3.6.3	Resumos . . . . .	39
3.7	Floats . . . . .	40
3.7.1	O Que São Floats . . . . .	40
3.7.2	O Ambiente figure . . . . .	40
3.7.3	Os Ambientes tabular e table . . . . .	40
3.7.4	Ambientes para Listagens . . . . .	40
<b>4</b>	<b>Controle de Referências</b>	<b>45</b>
4.1	O arquivo .bib . . . . .	45
4.1.1	Tipos de Entradas Mais Usados . . . . .	47
4.2	O Ecossistema $\text{\BibTeX}$ . . . . .	47
4.3	Comparando as Opções . . . . .	48
4.3.1	natbib vs biblatex . . . . .	48
4.3.2	$\text{\BibTeX}$ vs biber . . . . .	49
4.4	Usando biblatex e biber . . . . .	49
4.4.1	Exemplo dos comandos do biblatex . . . . .	50
4.5	Programas Externos . . . . .	50
4.5.1	JabRef . . . . .	51

# Resumo

Esse texto é uma introdução ao  $\text{\LaTeX}$  escrita em português, criada como resultado de um seminário de introdução ao  $\text{\LaTeX}$  realizado na época do Covid-19.

Ele não pretende ser a melhor introdução ao  $\text{\LaTeX}$ , apenas mais uma, mas foi criada com a intenção de facilitar um pouco a vida dos meus alunos.



# Capítulo 1

## Introdução ao Processamento de Texto

Os computadores foram criados para fazer contas, mas, na verdade, eles manipulam apenas símbolos. Rapidamente seus usuários perceberam que podiam ser usados para manipular textos, como código, e formatá-los de alguma forma para impressão. O processamento de texto é possivelmente a área de software com o maior número de usuários, já que todo usuário de computador, alguma vez, escreveu algo e enviou para alguém, nem que seja um simples e-mail. Para isso, foram criados vários tipos de programas, que cumprem funções como as descritas na figura 1.1, criando o que pode ser chamado de uma cadeia de processamento de texto. Basicamente, o que a figura mostra é um arquivo de texto pode ser editado, visualizado, processado para impressão ou processado de formas adicionais. Cada sistema de processamento de texto faz, prioritariamente uma dessas funções<sup>1</sup>.

### 1.1 Tipos de Sistema de Processamento de Texto

Dentro dessa cadeia de processamento existem vários tipos de programas. Os principais tipos são:

- Editores de Texto
- IDEs
- Processadores de Texto
- Sistemas de Autoria

---

<sup>1</sup>Observa-se que esta cadeia pode, para outros contextos, ser representada de forma mais complexa, incluindo conceito de produção, versionamento, segurança, etc.

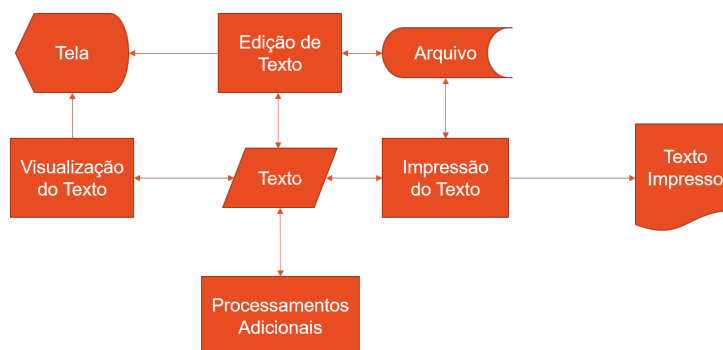


Figura 1.1: A cadeia de processamento de texto

- Sistemas de Publicação (*Desktop Publishing*)
- Sistemas de Composição
- Sistemas Colaborativos de Edição

### 1.1.1 Editores de Texto

Editores de texto são programas que permitem ao usuário editar arquivos que são de **texto puro**. Texto puro é um conceito que mudou. Inicialmente significava que havia um mapeamento um para um entre o que você encontrava no arquivo, uma sequência de caracteres codificados em ASCII<sup>2</sup>. Em ASCII, por exemplo, a letra “A” é representada pelos bits “1000001” e a “a” por “1100001”. Atualmente são usadas codificações que permitem que um caractere ou símbolo seja representado por uma sequência mais longa de bits, por meio de *escape codes*, por exemplo UTF-8<sup>3</sup>, o que significa que os editores de texto, normalmente, não representam mais perfeitamente o arquivo em disco. Os sistemas de codificação atuais são normalmente extensões do ASCII, isto é, os 128 códigos de 1 byte do ASCII original ainda são válidos. A figura 1.2 mostra a função de um editor de texto na cadeia de processamento de texto.

Editores de texto foram necessários assim que trocamos as entradas por cartão e fita, que eram editados em máquinas não conectadas ao computador, por terminais ligados diretamente aos mesmos. Os primeiros editavam linha a linha, a seguir outros exigiam que o usuário gerenciasse o *buffer*, o seja, a

<sup>2</sup>ASCII é um padrão que associa uma letra, e outros símbolos usados em arquivos, a um byte. Seu nome significa American Standard Code for Information Interchange. Baseado no alfabeto inglês, originalmente usava apenas 128 símbolos (7 bits), não possuindo os caracteres acentuados de outras línguas.

<sup>3</sup>Unicode Transformation Format, que permite codificar 1.112.064 símbolos



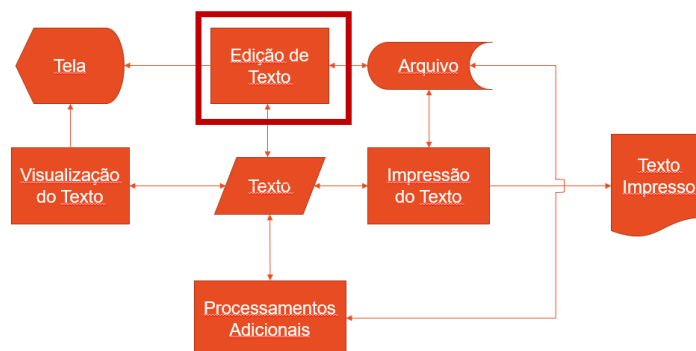


Figura 1.2: Função de um editor de texto na cadeia de processamento de texto.

parte do arquivo que estava em memória. Com o tempo chegamos a versões semelhantes as atuais.

Atualmente editores de texto tem um conjunto complexo de funções de busca, substituição, etc.

Exemplos de editores de texto atuais são o Notepad, que vem por *default* com o Windows, o *vi* e o *vim*, Notepad++, EditPlus, TextEdit e o poderosíssimo Emacs. A figura 1.3 mostra um exemplo to *vim*.

```

<!DOCTYPE html>
<html lang="en" xml:lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Vim Word Count and Useful Status Line</title>
    <meta name="description" content="Count the words in the file or
vim editor buffer, display details in a useful status line." />
    <meta name="keywords" content="Linux, Unix, vi, vim, editor, com
mand-line interface" />
    <?php @ include($_SERVER['DOCUMENT_ROOT'].'/ssi/header.html'); ?
  >
  <style>
    code,comment { color: #000080; }
  </style>
</head>
<body>
  <article itemscope itemtype="http://schema.org/Article" class="c
ontainer">
    <meta itemprop="author" content="Bob Cromwell" />
    <meta itemprop="about" content="Linux" />
    <header>
      //public-web/linux/vim-word-count.html[html] 1687 words, 6/363 lines, Top
  
```

Figura 1.3: O editor de texto vim

### 1.1.1.2 IDEs

Uma IDE, ou “Integrated Development Environment”, é uma extensão lógica da ideia de editor de texto criada originalmente para programadores, forne-

Uma IDE cobre a parte de edição e processamento de texto da cadeia de processamento de texto, como visto na figura 1.4.

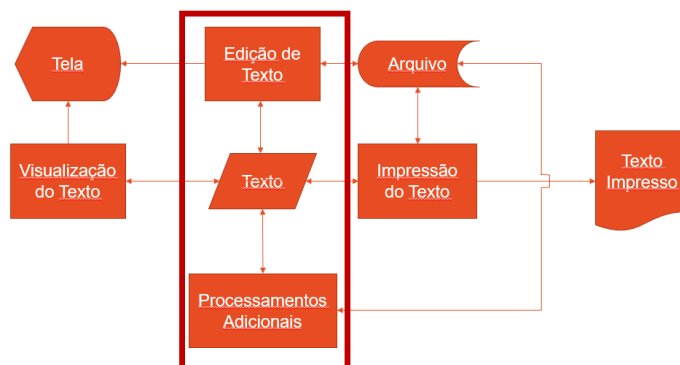


Figura 1.4: Um IDE permite editar e invocar outros processamentos de texto

Exemplos de IDE são o Atom, o Visual Studio, o T<sub>E</sub>XStudio e o próprio Emacs. Com a evolução dos editores de texto, a fronteira entre editores e IDEs ficou indefinida, muitas vezes dependendo do uso que o usuário faz do programa. A figura 1.5 mostra um exemplo do T<sub>E</sub>XStudio.

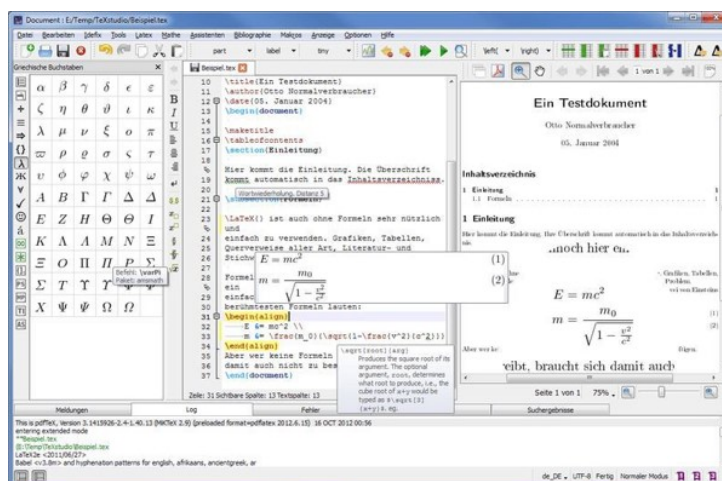


Figura 1.5: O T<sub>E</sub>X Studio

### 1.1.3 Sistemas de Composição (*Typesetting*)

Já que era possível editar textos, porque não imprimi-los de forma adequada? Essa ideia levou a criação de programas de tipografia, que faziam a tradução de um arquivo texto, com marcações adequadas, para um outro arquivo que fosse interpretado em uma impressora (ou outras máquinas mais sofisticadas de *typesetting* digital).

Um sistema de tipografia cobre apenas a parte de impressão da cadeia de processamento de texto, como visto na figura 1.6.

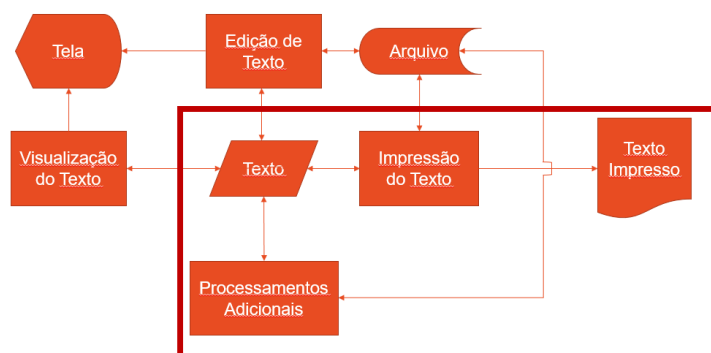


Figura 1.6: Sistemas de tipografia imprimem e fazem processamentos adicionais.

Essas marcações adequadas são como comandos, e um arquivo de texto marcado se assemelha a um programa de computador, por possuir palavras código que dão os comandos necessários. Sistemas desse tipo não possuem editores associados e são normalmente ativados por linha de comando.

Exemplos de sistemas de tipografia são o troff, o  $\text{\TeX}$  e o  $\text{\LaTeX}$ . A figura 1.7 mostra o  $\text{\LaTeX}$  sendo usado em linha de comando.

Os sistemas de composição mais avançados, como o  $\text{\LaTeX}$ , ou sistemas baseados em SGML, como o próprio HTML, tem a característica de separar a lógica do texto, sua estruturação, da sua forma de apresentação. Isso é conseguido no  $\text{\LaTeX}$  por meio do uso de classes de documentos, pacotes e comandos criados pelo usuário com essa intenção. No HTML isso é conseguido por meio do uso de classes para as tags e especificações CSS que definem a aparência das tags que pertencem aquela classe. Em XML é feito por meio de um processador externo que usa uma especificação que diz como o XML deve ser apresentado, por exemplo usando XSL como linguagem para definir o estilo a ser usado na apresentação.

```

F:\Github\Seminario-LaTeX>latexmk
Rc files read:
  .latexmkrc
Latexmk: This is Latexmk, John Collins, 17 Apr. 2020, version: 4.69a.
Latexmk: applying rule 'pdflatex'...
Rule 'pdflatex': The following rules & subrules became out-of-date:
  'pdflatex'
-----
Run number 1 of rule 'pdflatex'
-----
Running 'pdflatex -recorder "artigocomabstract.tex"'
-----
This is pdfTeX, Version 3.14159265-2.6-1.40.21 (MiKTeX 2.9.7400 64-bit)
entering extended mode
(arigocomabstract.tex
LaTeX2e <2020-02-02> patch level 5
L3 programming layer <2020-04-06>
("F:\Program Files\MiKTeX 2.9\tex\latex\base\article.cls"
Document Class: article 2019/12/20 v1.4l Standard LaTeX document class
("F:\Program Files\MiKTeX 2.9\tex\latex\base\size10.clo"))
("F:\Program Files\MiKTeX 2.9\tex\latex\base\fontenc.sty")
("F:\Program Files\MiKTeX 2.9\tex\generic\babel\babel.sty"
("F:\Program Files\MiKTeX 2.9\tex\generic\babel\babel.def"
("F:\Program Files\MiKTeX 2.9\tex\generic\babel\xtbabel.def")
*****
* Local config file bbltops.cfg used
*
("F:\Program Files\MiKTeX 2.9\tex\latex\arabi\bbltops.cfg")
("F:\Program Files\MiKTeX 2.9\tex\latex\babel-english\english.ldf")

```

Figura 1.7: O  $\text{\LaTeX}$  sendo usado

### 1.1.4 Processadores de Texto

Principalmente com o advento do micro-computador, ficou claro que uma das principais utilidades do computador seria permitir a criação de textos a serem publicados, logo um editor de texto deveria ser estendido para suportar funções como colocar palavras em negrito, itálico, selecionar fontes, etc.

Processadores de texto cumprem grande parte das funções na cadeia de processamento de texto, como mostra a figura 1.8.

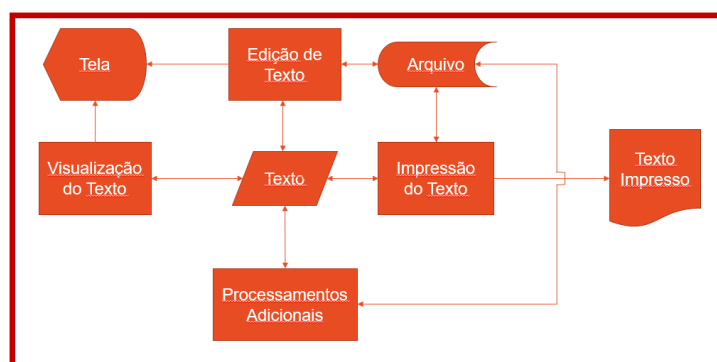


Figura 1.8: Os processadores de texto na cadeia de processamento de texto.

Com a evolução dos terminais de computadores, micro-computadores e placas de vídeo e monitores, os processadores de texto evoluíram de sistemas

que mostravam alguma coisa do que estava sendo prevista para o texto, como caracteres em bold, para sistemas que permitiam uma visualização prévia, até chegar a edição pelo conceito de WYSIWYG, *What You See Is What You Get*, lido “uiziwig”, que mostra na tela quase que exatamente o que será visto na edição final, sendo as diferenças mínimas e quase imperceptíveis causadas por questões tecnológicas e da diferença entre papel e tintas e monitores.

Atualmente o Word é o processador de texto que domina amplamente o mercado, e seus principais concorrentes são sistemas de código aberto, como o LibreOffice Writer ou o Apache OpenOffice Writer. Um tela do Word é mostrada na figura 1.9.

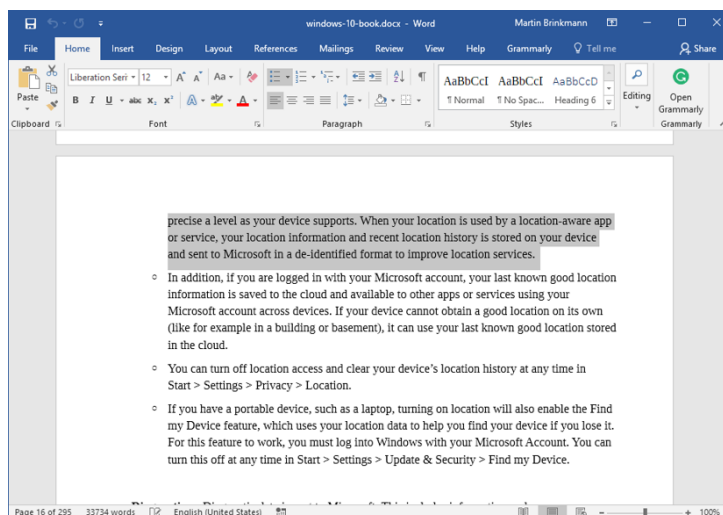


Figura 1.9: O Word é o principal processador de texto do mercado.

Uma característica avançada importante dos processadores de texto é o uso de estilos, e templates, que são basicamente coleções de estilos organizadas de modo a dar um formato coerente ao documento. Por meio do uso de estilos é possível garantir que partes do documento que tem a mesma função tenham a mesma aparência.

### 1.1.5 Sistemas de Autoria

Sistemas de autoria são uma evolução interessante dos processadores de texto, ou dos IDEs, voltadas para autores de livros, roteiros, etc. que não só permitem editar o texto, em formatos específicos, como também guardar informações como fichas de personagens, descrições de cena, *storyboards*, etc... Eles podem cobrir toda a cadeia de processamento de texto.

## 14 CAPÍTULO 1. INTRODUÇÃO AO PROCESSAMENTO DE TEXTO

Mesmo o mercado não sinalizando isso, sistemas de autoria tem grande potencial para o futuro e para a academia, pois normalmente ao escrever algo precisamos guardar muito informação relativa ao conteúdo, e um sistema de autoria poderia ajudar a fazê-lo.

Exemplo de sistemas de autoria são o Scrivener, apresentado na figura 1.10, o Final Draft e o Celtx.

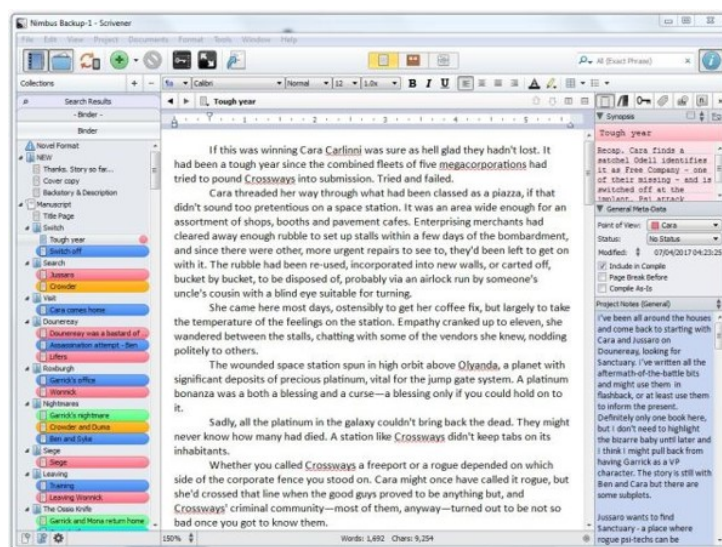


Figura 1.10: Uma tela do Scrivener

### 1.1.6 Sistemas de Publicação

Sistemas de publicação, ou *Desktop Publishing*, são sistemas cujo o foco é a diagramação de publicações, fornecendo uma capacidade menor de processamento de texto. São nesses sistemas que a maioria de jornais e revistas, e também manuais, folhetos, e outras formas de documentos impressos, são hoje preparados para a impressão.

Exemplos típicos de sistemas de publicação são o Publisher e o Frame-maker, que é mostrado na figura 1.11.

Sistemas de publicação são mais difíceis de usar que sistemas de processamento de texto, porém permitem fazer a diagramação de um documento de formas mais avançadas, e depois colocar os textos nos espaços dentro da diagramação. Normalmente o texto é editado em outros programas, por exemplo, em um processador de texto, que permite colocar já os formatos como negrito e itálico.

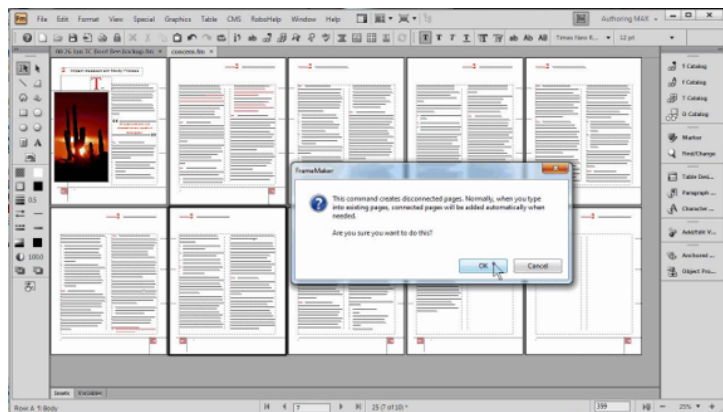


Figura 1.11: O software Framemaker

### 1.1.7 Sistemas Colaborativos de Edição

Esses sistemas aparecem cedo, porém se expandem principalmente com o fortalecimento da internet. Eles permitem que mais de uma pessoa edite um arquivo ao mesmo tempo. Atualmente o mais conhecido é o Google Docs, que é um processador de texto limitado em funcionalidade mas muito fácil de usar.

O ShareLateX foi um sistema colaborativo de edição voltado para o  $\text{\LaTeX}$  que acabou sendo responsável por um renascimento do uso do  $\text{\LaTeX}$  na academia. Acabou sendo incorporado ao concorrente Overleaf, que aparece na figura 1.12.

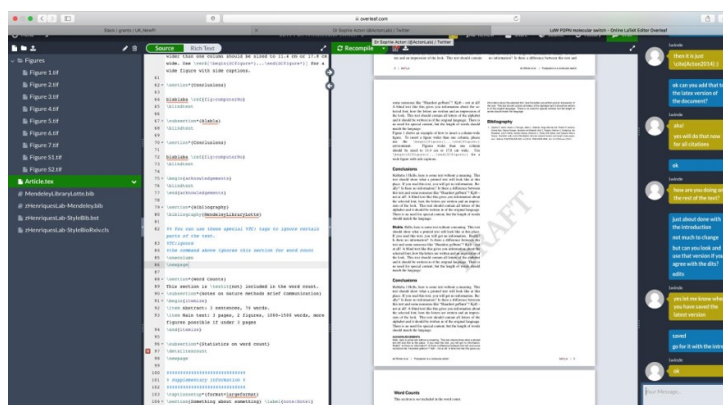


Figura 1.12: O Overleaf.



## 1.2 Tipos de Linguagens para Arquivos de Texto

Arquivos de texto podem ser guardados de várias formas. No limite, podemos fotografar uma página de texto e guardar a imagem, mas geralmente queremos um arquivo que possa ter seu texto manipulado.

Normalmente, na cadeia de processamento de texto, o arquivo é lido e colocado em memória em um formato mais adequado, e depois, quando salvo, é “rearrumado” de alguma forma, um formato de arquivo texto. Por exemplo, ao salvar um arquivo Word você pode escolher o seu formato nativo (.docx) ou vários outros formatos alternativos, como RTF<sup>4</sup>, ou mesmo um arquivo texto, nesse caso perdendo toda a formatação.

Os formatos de arquivo podem ser divididos em:

- Linguagens de Impressão/Visualização
  - PostScript, DVI, PDF
- Linguagens Intermediárias (de impressão)
  - .dvi
- Linguagens de Marcação
  - SGML, HTML,  $\text{\TeX}$ ,  $\text{\LaTeX}$  Markdown, RPF, fods

O fato de uma linguagem de marcação ser legível por humanos não quer dizer que seja facilmente legível, principalmente quando geradas por máquinas. A figura 1.13 mostra um exemplo de arquivo Postscript.

```
%!PS-Adobe-3.0
%%Title: Datamatrix Barcode
%%Creator: JpGraph Barcode http://www.aditus.nu/jpgraph/
%%CreationDate: Sun 5 Jul 23:06:27 2009
%%DocumentPaperSizes: A4
%%EndComments
%%BeginProlog
%%EndProlog

%%Page: 1 1

%%Module width: 3 pt

%%Data for bars. Only black bars are defined.
%%The figures are for each row and in format: [xpos]
%%Data: A Datamatrix barcode
3.05 setlinewidth
[00] [6] [12] [18] [24] [30] [36] [42] [48] [54] [57] {} forall 60 moveto 0 -3.05 rlineto stroke forall
[00] [6] [12] [18] [21] [33] [36] [39] [42] [51] [54] [57] [57] {} forall 57 moveto 0 -3.05 rlineto stroke forall
[00] [18] [21] [24] [36] [42] [51] [51] {} forall 54 moveto 0 -3.05 rlineto stroke forall
[00] [12] [15] [24] [30] [36] [39] [42] [45] [48] [51] [54] [57] [57] {} forall 51 moveto 0 -3.05 rlineto stroke forall
[00] [6] [9] [21] [24] [33] [36] [51] [54] [57] {} forall 48 moveto 0 -3.05 rlineto stroke forall
[00] [3] [6] [15] [18] [21] [24] [33] [48] [51] [57] [57] {} forall 45 moveto 0 -3.05 rlineto stroke forall
[00] [9] [12] [15] [24] [27] [39] [51] [54] [57] {} forall 42 moveto 0 -3.05 rlineto stroke forall
[00] [6] [18] [24] [27] [33] [39] [42] [48] [54] [57] [57] {} forall 39 moveto 0 -3.05 rlineto stroke forall
[00] [9] [12] [15] [21] [24] [27] [30] [33] [39] [45] [54] [57] {} forall 36 moveto 0 -3.05 rlineto stroke forall
[00] [3] [15] [18] [21] [24] [30] [33] [36] [39] [48] [57] [57] {} forall 33 moveto 0 -3.05 rlineto stroke forall
[00] [3] [6] [18] [21] [27] [39] [45] [57] {} forall 30 moveto 0 -3.05 rlineto stroke forall
[00] [15] [18] [21] [27] [30] [33] [36] [42] [51] [54] [57] [57] {} forall 27 moveto 0 -3.05 rlineto stroke forall
[00] [3] [6] [12] [15] [18] [21] [30] [33] [36] [39] [42] [45] [48] [57] {} forall 24 moveto 0 -3.05 rlineto stroke forall
[00] [6] [12] [27] [30] [42] [57] [57] {} forall 21 moveto 0 -3.05 rlineto stroke forall
[00] [3] [9] [15] [18] [24] [33] [39] [42] [45] [48] [57] {} forall 18 moveto 0 -3.05 rlineto stroke forall
[00] [9] [12] [21] [27] [30] [36] [39] [42] [45] [57] [57] {} forall 15 moveto 0 -3.05 rlineto stroke forall
[00] [9] [12] [24] [39] [42] [48] [51] [54] [57] {} forall 12 moveto 0 -3.05 rlineto stroke forall
[00] [9] [21] [27] [36] [39] [42] [45] [51] [57] [57] {} forall 9 moveto 0 -3.05 rlineto stroke forall
[00] [15] [21] [24] [27] [36] [39] [42] [45] [54] [57] {} forall 6 moveto 0 -3.05 rlineto stroke forall
[00] [3] [6] [9] [12] [15] [18] [21] [24] [27] [30] [33] [36] [39] [42] [45] [48] [51] [54] [57] [57] {} forall 3 moveto 0 -3.05 rlineto stroke forall

%%End of Datamatrix Barcode
showpage
%%Trailer
```

Figura 1.13: Exemplo de um arquivo PostScript

---

<sup>4</sup>Rich Text Format





Figura 1.14: Mapa mental das linguagens de marcação.

### 1.2.1 Linguagens de Marcação

Um linguagem de marcação é caracterizada por um conjunto de códigos que é aplicado sobre o texto, ou sobre qualquer forma de dados, com o fim de adicionar informações específicas sobre esse texto, a cada trecho específico, como por exemplo a forma de exibi-lo graficamente.

Linguagens de marcação são uma forma simples de indicar como um texto deve ser impresso, e são usadas desde o início da digitalização do processo de composição e impressão.

As linguagens de marcação padronizadas permitem que programas diversos cumpram a mesma função. Elas podem ser divididas em:

- Procedurais
  - troff,  $\text{\TeX}$ ,  $\text{\LaTeX}$ , Postscript
- De apresentação
  - Wikis, Markdown
- Descritivas
  - SGML, HTML, XML

Linguagens de marcação procedurais normalmente incluem instruções de composição, ou seja, elas mantêm unificadas a estrutura e a apresentação, enquanto linguagens de apresentação não se preocupam com nada além da apresentação imediata. Já linguagens descritivas, introduzidas por Scribe, separam a estrutura da apresentação.  $\text{\LaTeX}$ , apesar de ainda conter instruções de composição, foi fortemente influenciada por essa ideia, e o usuário final praticamente só usa instruções que falam sobre a estrutura do documento. Um mapa mental das linguagens pode ser visto na figura 1.14(Adams, 2007).

Linguagens de marcação normalmente são criadas para serem simultaneamente compreensíveis para o ser humano e tratáveis por um programa de

computador.

Um exemplo de código  $\text{\LaTeX}$  e seu resultado é mostrado na figura 1.15.

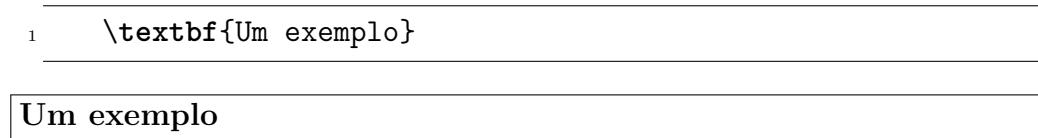


Figura 1.15: Exemplo de código  $\text{\LaTeX}$

### 1.2.2 Linguagens de Impressão

Linguagens de impressão tem como finalidade servir apenas para o processamento por um mecanismo de impressão ou um software específico de visualização. Nesse caso, não há preocupação com a compreensão do formato por seres humanos.

Exemplos de linguagens de impressão são o formato PDF, que é um Postscript empacotado de forma a descrever documentos compostos de páginas.

$\text{\TeX}$  trouxe o conceito de uma linguagem de impressão intermediária, por meio do formato DVI, que torna o arquivo gerado pelo  $\text{\TeX}$  independente do processamento final, o que permite que cada fabricante, ou interessado em geral, faça um programa próprio de conversão de DVI para qualquer formato de impressão. Logo, documentos DVI podem ser transformados em PDF, PS ou outro formato por software especiais, mas atualmente as implementações de  $\text{\LaTeX}$  tornam isso transparente.

## 1.3 O que é o $\text{\LaTeX}$

$\text{\LaTeX}$  é um sistema de composição, ou *typesetting*, baseado no  $\text{\TeX}$  e apoiado com outros programas, como o biber, que permite usar arquivos de texto marcados para criar arquivos a serem impressos, possivelmente no formato PDF, seguindo regras de composição e usando fontes detalhadamente criadas para reproduzir a qualidade de fontes utilizada na composição manual, incluindo especialmente as ligaduras, que são caracteres especiais que representam dois ou mais caracteres de forma visualmente mais elegantes, como mostradas na figura 1.16.

$fi$	$\longrightarrow$	$f_i$	$AE$	$\longrightarrow$	$\mathcal{A}E$
$ff$	$\longrightarrow$	$ff$	$ae$	$\longrightarrow$	$\mathcal{a}e$
$ffi$	$\longrightarrow$	$ffi$	$OE$	$\longrightarrow$	$\mathcal{O}E$
$fl$	$\longrightarrow$	$fl$	$oe$	$\longrightarrow$	$\mathcal{o}e$
$\dot{ij}$	$\longrightarrow$	$\ddot{y}$	$LATEX$	$\longrightarrow$	$\text{\LaTeX}$

Figura 1.16: Exemplos de ligaturas de fontes do  $\text{\TeX}$ .

## 1.4 Por que $\text{\LaTeX}$ ?

Alguns argumentos dados para usar  $\text{\LaTeX}$  são:

- qualidade estética, os defensores de  $\text{\LaTeX}$  dizem que por ser um sistema de composição, baseado em algoritmos que buscam soluções ótimas segundo regras específicas e muitas vezes desconhecidas pelos autores;
- preço,  $\text{\LaTeX}$  é grátis;
- o documento `.tex` é independente de fabricante, pode ser editado com qualquer editor e ser usado plenamente com gerenciadores de versão. Você sempre poderá editar um arquivo `.tex`;
- torna fácil trocar o formato de arquivos grandes, trocando apenas o estilo;
- WYSIWYG é na verdade um What You See Is All You've Got, não sendo possível modificar o comportamento além do que o sistema está oferece para você<sup>5</sup>;
- foco no conteúdo, deixando as questões de formato para os criadores de estilo, e
- longevidade, está em utilização ampla há mais de anos.

Alguns problemas de  $\text{\LaTeX}$  são:

- esforço de tempo para aprender  $\text{\LaTeX}$ ;
- dificuldade de resolver alguns erros;
- dificuldade de criar um estilo novo como você deseja, e
- as alterações não aparecem imediatamente, tendo que ser compiladas.

---

<sup>5</sup>Isso fica mais claro se você usar um sistema WYSIWYG mais limitado como o Google Docs

### 1.4.1 O Design Lógico de Documentos

A ideia do design lógico de documentos, proposta inicialmente por Reid (1980) defende que o autor não deve se preocupar com o formato enquanto escreve o documento. Isso é exatamente o oposto que os sistemas WYSIWIG induzem, já que sempre estamos vendo o que seria o formato final.

## 1.5 Sistemas Mais Usados na Computação

Três sistemas são hoje muito usados na Computação:

- **Word**
  - Um software WYSIWYG
  - Líder do mercado
  - Superpoderoso
  - Difícil de usar para trabalho colaborativo
- **Google Docs**
  - Quase WYSIWYG
  - Sucesso entre os jovens
  - Pouca capacidade de plena expressão gráfica
  - Ótimo para trabalho colaborativo
- **L<sup>A</sup>T<sub>E</sub>X**
  - Melhor imagem de texto, mas no detalhe
  - Ótimo para Matemática
  - Difícil de usar
  - Empoderado pelo Overleaf
  - Renovado com os sistemas colaborativos

Desses sistemas, claramente o Google Doc ainda não serve para gerar documentos com formatação de qualidade, ou documentos destinados a artigos de revistas ou livros. Então ficaremos, na próxima seção, apenas com a discussão dos mitos e fatos que aparecem na discussão do uso de Word vs L<sup>A</sup>T<sub>E</sub>X.

Chamamos a atenção que não consideramos viável o uso para objetivos acadêmicos dos sistemas abertos, como Open Office ou LibreOffice, por possuírem problemas de compatibilidade dos resultados de visualização entre versões dos próprios e com formatos disponíveis para o Word<sup>6</sup>.

---

<sup>6</sup>Será isso um mito? Não é o que parece pela experiência do autor

## 1.6 Mitos e Fatos de Word vs L<sup>A</sup>T<sub>E</sub>X

### 1.6.1 L<sup>A</sup>T<sub>E</sub>X é mais produtivo

A suposição seria que, como você não se preocupa com o resultado final, já que ele é controlado pela classe de documento e estilos. Isso é falso, e já foi comprovado em pesquisas científicas que a produtividade é igual ou menor.

### 1.6.2 A qualidade de saída do L<sup>A</sup>T<sub>E</sub>X é muito melhor

**Verdade**, mas possivelmente só para quem entende o que está vendo. A qualidade da saída proporcionada pelo T<sub>E</sub>X depende de algum conhecimento do que é uma boa fonte, uma boa distribuição de texto. É possível que um leitor tenha alguma sensação de beleza ou conforto superior quando vê um texto gerado pelo T<sub>E</sub>X, mas é possível também que não saiba diferenciar, pois são detalhes.

### 1.6.3 Word não trabalha bem com fórmulas matemáticas

**Falso**. A nova maneira de trabalhar com fórmulas do Word, que já está funcionando há algum tempo, não só é muito boa como permite escrever as fórmulas com a mesma sintaxe do que o L<sup>A</sup>T<sub>E</sub>X.

Ainda há a questão da beleza das fórmulas, mas a diferença também é pequena hoje em dia.

### 1.6.4 Você perde muito tempo com besteira no L<sup>A</sup>T<sub>E</sub>X

**Verdadeiro**. Como L<sup>A</sup>T<sub>E</sub>X é uma linguagem compilada, você pode perder muito tempo com erros espúrios. Além disso, a capacidade de detectar o erro no momento certo é fraca no L<sup>A</sup>T<sub>E</sub>X, e coisas como parênteses faltando podem causar mensagens de erro muito longe. Porém, com o uso, isso tende a diminuir.

### 1.6.5 Não consigo colocar a imagem onde quero no Word

**Falso**. Sinceramente, nem sei de onde vem esse mito. Você pode colocar a imagem onde quiser no Word. O mesmo, com o L<sup>A</sup>T<sub>E</sub>X, já é mais difícil.

### 1.6.6 Em $\text{\LaTeX}$ gerencio melhor as bibliografias

**Falso.** Na verdade,  $\text{\LaTeX}$  não gerencia a bibliografia, ele implica no uso de outro sistema, o  $\text{\BibTeX}$ , por exemplo, e outro processador, como o `biber`. O mesmo pode ser feito com o Word, usando sistemas como Zotero ou Paper.

### 1.6.7 Em $\text{\LaTeX}$ consigo controlar versões

**Falso, parcialmente.** Este autor é um grande fã do uso de gerência de versões, sendo um usuário do Git e do GitHub. Realmente é fácil fazê-la com o  $\text{\LaTeX}$ , porque todos os arquivos são textuais e plenamente controlados no controle de versões, porém um efeito quase similar para a visualização de diferenças pode ser obtido com a instalação do software `pandoc` e uma pequena configuração do Git. Outras funcionalidades, como o *merge* de versões com conflito não são possíveis.

### 1.6.8 Word é mais fácil de aprender

**Verdade,** porém o usuário médio não usa corretamente as funções que tornam o software Word um programa fantástico, como o uso de templates e estilos.

### 1.6.9 Com o Word, basta ele

**Falso.** Para fazer tudo que você faz com o  $\text{\LaTeX}$ , você precisa de outros programas para o Word também.

### 1.6.10 Word tem problemas com arquivos grandes

**Verdade, para arquivos muito grandes,** o Word realmente às vezes se perde e causa bugs, quase impossíveis de consertar, com arquivos muito grandes. A alternativa de usar arquivos *master* que incluem outros arquivos, o que se faz com facilidade no  $\text{\LaTeX}$ , está quebrada há várias versões e não há perspectiva de ser corrigida<sup>7</sup>.

## 1.7 Recomendações

Essas são as minhas recomendações de quando usar que sistema:

- **Word**

---

<sup>7</sup>Sim, isso é algo bizarro de se saber.

- Ótima opção para um autor e um revisor
- Bom para arquivos pequenos e grandes, não use para arquivos muito grandes.
- Separe o texto em capítulos, e só junte na hora de imprimir.
- Cuidado com o backup, faça sempre
- Se possível, use o controle de versões
- Aprenda a usar estilos, principalmente de parágrafos, e templates
- Não separe parágrafos com linha em branco e não comece parágrafos com tab, use estilos para isso.
- **Google Docs**
  - Ótimo para colaborações onde pessoas escrevem ao mesmo tempo
  - Sem o controle de quem fez o que, o que é possível com controle de versão (opção *Blame* no GitHub)
  - Funciona bem com arquivos pequenos, não foi testado com arquivos grandes ou enormes
- **L<sup>A</sup>T<sub>E</sub>X**
  - Se a editora fornece o formato (.sty) é a melhor opção
  - Para exames, dissertações e teses da Coppe é a melhor opção
    - ◊ Use o formato CoppeTeX disponíveis em: <https://github.com/COPPE-UFRJ/CoppeTeX>
    - ◊ basta baixar todo o diretório dist
  - Use o JabRef ou o Zotero com Better BibTeX for Zotero
  - L<sup>A</sup>T<sub>E</sub>X no Overleaf
    - ◊ Bom para artigos
    - ◊ Pode não conseguir compilar uma tese, pois tem limite de tempo.
    - ◊ Ligue as opções GitHub e Dropbox, e faça backup
  - L<sup>A</sup>T<sub>E</sub>X em casa
    - ◊ Use o Git+Nuvem (GitHub, GitLab, etc.)
    - ◊ Mantenha a instalação atualizada
    - ◊ Faça backup





# Capítulo 2

## O Mundo L<sup>A</sup>T<sub>E</sub>X

Nesse capítulo é feita uma introdução aos conceitos que giram em torno do L<sup>A</sup>T<sub>E</sub>X.

### 2.1 Invenção do L<sup>A</sup>T<sub>E</sub>X

No começo, não existiam computadores. Livros eram escritos a mão ou dactilografados e um tipógrafo os montavam por meio de tipos móveis, página a página, para imprimi-los, primeiro como um grande carimbo como as máquinas de Gutenberg, depois por processos mais sofisticados, como o linotipo ou o offset.

Ao escrever o primeiro volume da série seminal de livros *The Art of Computer Programming*, Donald Knuth, que ganhou o prêmio Turing de 1974, teve o livro feito da forma clássica, por composição a quente. No segundo volume, em 1969, foi utilizada uma composição digital. Seu desagrado com o resultado visual o levou a um projeto de 10 anos em tipografia digital, período em que criou o T<sub>E</sub>X o METAFONT, um método de programação conhecido como *Literate Programming* e os programas WEB e CWEB que o implementam.

Um texto processado em T<sub>E</sub>X aparece na listagem 2.1, e seu resultado na figura 2.1.

---

Listagem 2.1: Exemplo de arquivo em T<sub>E</sub>X puro.

---

```
1 % exemplo de comentário
2 \magnification\magstep1
3 \font\logo=logo10 % font used for the METAFONT logo
4 \def\MF{\logo META}\-{\logo FONT}}
5 \def\TeX{T\hbox{\hskip-.1667em\lower.424ex\hbox{E}}
6 \hskip-.125em X}}
7 \def\bib{\par\noindent\hangindent 20pt}
```

```

8 \line{\bf Um exemplo de arquivo \TeX\ \ \ \hfill}
9 \bigskip\noindent
10 Este texto é um exemplo simples de como pode ser
11 gerado um arquivo \TeX.
12 A linguagem permite
13 usar comando e criar macros, tendo sido usada para escrever
14 {\sl The Art of Computer Programming}.
15 \bye

```

---

Figura 2.1: Resultado do exemplo do uso do  $\text{\TeX}$ puro.

Deve ficar claro ao leitor que a linguagem  $\text{\TeX}$  é uma linguagem de marcação orientada a como vai ser a visualização do texto. Já na década de 80, influenciado pelas propostas de descrever um texto pela sua lógica, a proposta pelo sistema Scribe (Reid, 1980), separando a visualização, Leslie Lamport, que recebeu o prêmio Turing de 2013, escreveu algumas macros para si, que foram distribuídas para colegas, até que foi convidado a escrever um livro que as descrevesse. Assim nasceu  $\text{\LaTeX}$  (Mittelbach et al., 1999). A versão atual do livro se chama  *$\text{\LaTeX}$ (2nd Ed.): A Document Preparation System: Users Guide and Reference Manual* (Lamport, 1994).

É importante notar que os comportamentos previstos tanto do  $\text{\TeX}$  quanto do  $\text{\LaTeX}$  são bastante estáveis, porém o  $\text{\LaTeX}$  sobre evolução constante, tanto de seu funcionamento básico como das centenas de pacotes disponíveis na CTAN.

## 2.2 Situação Atual do $\text{\LaTeX}$

As implementações mais conhecidas de  $\text{\LaTeX}$  e os programas auxiliares são:

- Várias implementações de  $\text{\TeX}$ 
  - $\text{\pdfTeX}$  – processador que ficou sendo o mais usado
  - $\text{\XeTeX}$  – Unicode + novas fontes
  - $\text{\LuaTeX}$  – evolução do  $\text{\pdfTeX}$  onde todas as chamadas internas podem ser acessadas por Lua

- Programs relacionados
  - $\text{\BibTeX}$  – programa original de tratamento de citações e bibliografia
  - $\text{\biber}$  – processador de referências mais moderno, para usar o  $\text{\bibTeX}$ .
  - $\text{\JabRef}$  – gerenciador de referências
- Editores
  - $\text{\TeXnicCenter}$
  - $\text{\TeXStudio}$
  - $\text{\TeXMaker}$

Diferentes distribuições incluem diferentes versões de  $\text{\TeX}$ ,  $\text{\LaTeX}$  e os programas relacionados. As principais distribuições são:

- Multi-sistema
  - $\text{\TeXLive}$  – A principal distribuição, com dezenas de desenvolvedores
- No Windows
  - $\text{\MiKTeX}$  – Uma distribuição ativamente mantida e usando *wizards* para instalação, preferida por muitos usuários Windows por causa dessa facilidade
  - $\text{\proTeXt}$  –  $\text{\MiKTeX}$  com alguns adicionais
- No Linux –
  - Pacote disponível na distribuição, geralmente uma versão do  $\text{\TeXLive}$ , normalmente de atualização mais lenta.
- No Mac
  - $\text{\MacTeX}$  –  $\text{\TeXLive}$  especializada para o Mac

Também é possível usar o  $\text{\LaTeX}$  na nuvem, nos seguintes sites:

- Overleaf
  - <https://www.overleaf.com/>
  - O principal ambiente de edição compartilhada de  $\text{\LaTeX}$ , principalmente depois de comprar o ShareLaTeX
- Papeeria
  - Permite editar  $\text{\LaTeX}$  e Markdown, baseado no  $\text{\TeXLive}$
  - <https://papeeria.com/>
- Cocalc
  - <https://cocalc.com/doc/latex-editor.html>

As principais informações sobre  $\text{\LaTeX}$  e  $\text{\TeX}$  podem ser obtidas em:

- CTAN Comprehensive TEX Archive Network: <https://ctan.org/>
- $\text{\TeX}$  Stack Exchange : <https://tex.stackexchange.com/>

- $\text{\TeX}$  FAQ <https://texfaq.org/>
- $\text{\TeX}$  User Group TUG: <https://www.tug.org/>
- The  $\text{\LaTeX}$  Project: <https://www.latex-project.org/>

## 2.3 Como funciona o $\text{\LaTeX}$

Na prática, o  $\text{\LaTeX}$  funciona como um compilador. Sua entrada principal é um arquivo `.tex`, que contém o texto do documento a ser impresso. Na verdade, vários arquivos são lidos, muitos instalados na distribuição e fora da visão imediata do autor, sendo os principais:

- um arquivo `.cls`, que define a classe do documento;
- zero ou mais arquivos `.sty`, que são as implementações dos pacotes utilizados;
- arquivos que definem como será tratada a bibliografia, como `.bbx`, `.cbx` ou `.ltx`, que são parte de pacotes de bibliografia e chamados pelo pacote;
- arquivos com a bibliografia propriamente tida, `.bbl` no caso do biber estar sendo usado;
- arquivos `.tex` incluídos pelo autor a partir do documento raiz, e
- arquivos de imagem, `.png`, `.jpg`, `.pdf` ou outros, incluídos pelo autor a partir do documento raiz.

Com o uso de citações e bibliografia, o  $\text{\LaTeX}$  precisa fazer o processamento pelo menos 3 vezes: a primeira para gerar um arquivo de entrada para o biber, que é então executado, a segunda para colocar o que o biber gerou em seu lugar correto e a terceira para ajustar todas as referências de acordo com a configuração final do documento. Esse processo, que é apenas uma descrição parcial do que é mais visível ao autor, é representado na figura 2.2.

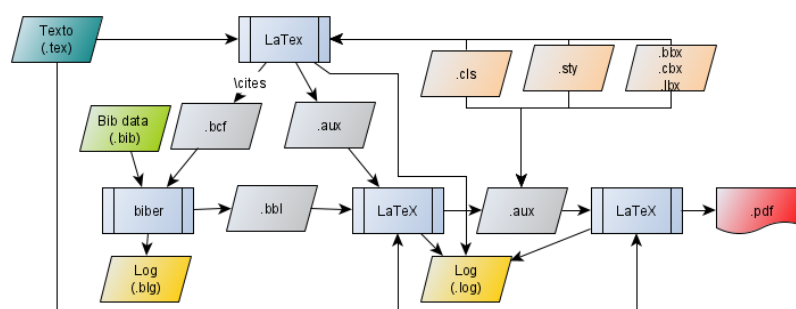


Figura 2.2: O fluxo de processamento (simplificado) do  $\text{\LaTeX}$

## 2.4 Recomendações de uso

- Comece pelo Overleaf, até ter interesse de mudar para sua máquina
- Sincronize o Overleaf via Dropbox w=e GitHub.
- No PC
  - Comece com o MiKTeX, é muito mais fácil de instalar e manter atualizado;
- No Linux
  - Use o T<sub>E</sub>XLive em vez da distribuição padronizada
- No Mac
  - Aparentemente, sua única opção é o MacT<sub>E</sub>X.
- Use o T<sub>E</sub>X Studio para editar.
- Para processar, use o LuaT<sub>E</sub>X e LuaL<sup>A</sup>T<sub>E</sub>X
- Para bibliografia, use o BibL<sup>A</sup>T<sub>E</sub>X (forma do biber)
  - Cuidado com estilos que exigem o B<sub>B</sub>T<sub>E</sub>X
  - Há diferenças sutis de comportamento
- Sempre mantenha seu projeto L<sup>A</sup>T<sub>E</sub>X no Git
  - GitHub, GitLab e etc.



# Capítulo 3

## L<sup>A</sup>T<sub>E</sub>X Básico

### 3.1 L<sup>A</sup>T<sub>E</sub>X Muito Básico

#### 3.1.1 Documento Mínimo

Um documento mínimo L<sup>A</sup>T<sub>E</sub>X tem a aparência da listagem 3.1. Sua aparência é a da figura 3.1

Listagem 3.1: Um documento mínimo em L<sup>A</sup>T<sub>E</sub>X.

---

```
1 \documentclass[a4paper]{article}
2 % arquivo semi1.tex
3 %isto é um comentário
4 %preâmbulo
5 %onde colocamos comandos
6 \begin{document}
7   %área de texto
8   Hello World!
9 \end{document}
```

---

Cada documento L<sup>A</sup>T<sub>E</sub>X começa com a declaração de que classe ele usa. A classe de um documento é sua principal guia de formatação/composição, sendo completada por pacotes, que não foram usados no exemplo mínimo. Cada classe pode possuir opções, como a opção **a4paper** usada no exemplo,

Hello World!

Figura 3.1: Texto que será impresso em uma página após o processamento do arquivo da listagem 3.1

e que define o tamanho do papel.

O símbolo de porcentagem indica o início de um comentário. Um arquivo *L<sup>A</sup>T<sub>E</sub>X* tem duas áreas principais, o preâmbulo, onde são colocados os comandos de configuração da saída, e o documento propriamente dito, onde é colocado o texto.

O formato básico de um comando *L<sup>A</sup>T<sub>E</sub>X* é:

`\comando[<opções>]{<parâmetro>}`

porém existem comandos com colchetes após os parâmetros ou mais de um parâmetro, como veremos mais adiante.

Neste texto vamos usar a classe `article`, porém é muito comum que as pessoas usem outras classes, como classes fornecidas pela editora que vai publicar seu artigo ou livro. Classes bastante usadas são:

- `article`, genérica para artigos
- `report`, genérica para relatórios técnicos
- `book`, genérica para livro
- `IEEEtran`, classe para publicações diversas da IEEE

Uma das facilidades fornecidas por *L<sup>A</sup>T<sub>E</sub>X* é escrever seu artigo em uma classe genérica e depois simplesmente trocar a classe para a da editora desejada.

Existe uma classe para os documentos acadêmicos da Coppe, mantida por um grupo de voluntários que inclui este autor, e que cobre dissertação, tese, exame de qualificação e outras coisas. Ela pode ser obtida no GitHub no endereço <https://github.com/COPPE-UFRJ/CoppeTeX/tree/master/dist>

## 3.2 Estrutura de um documento

Um documento *L<sup>A</sup>T<sub>E</sub>X* utiliza uma estrutura de partes hierárquicas, que, nos principais estilos são:

- `\part{<título>}` – só para `report` e `book`
- `\chapter{<título>}` – só para `report` e `book`
- `\section{<título>}`
- `\subsection{<título>}`
- `\subsubsection{<título>}`
- `\paragraph{<texto>}`
- `\subparagraph{<texto>}`

A listagem 3.2 mostra um documento usando a classe `article` e dividido em seções e subseções. Parágrafos não marcados são separados por uma



linha vazia. Uma prática comum em  $\text{\LaTeX}$  é preencher um frase, da letra maiúscula ao ponto final, por linha de texto, e, obviamente, manter as linhas de um parágrafo juntas.

Apesar dos exemplos estarem gerando páginas de PDF, como mostrado na figura 3.2 a maior parte das figuras será cortada para evitar excesso de espaço em branco neste documento.

Listagem 3.2: Um artigo básico em  $\text{\LaTeX}$ .

---

```
1 \documentclass{article}
2 \usepackage[T1]{fontenc}
3 \usepackage[english,brazilian]{babel}
4
5 \title{Meu Primeiro Artigo}
6 \author{Geraldo Xexéo}
7
8 \begin{document}
9
10 \maketitle
11
12 \section{A primeira seção}
13 Lorem ipsum dolor sit amet, consectetur
14 adipiscing elit. Duis ultricies efficitur aliquet.
15
16
17 \section{A segunda seção}
18 Nulla iaculis bibendum sapien.
19 Donec sollicitudin pharetra ipsum euismod vulputate.
20
21 \subsection{Uma subseção}
22 Sed ut nisl lectus.
23
24 Integer faucibus, est eu mattis vestibulum
25 \end{document}
```

---

### 3.3 Informação de Capa

Algumas informações que devem existir em um documento, que chamaremos de informação de capa ou de título, são colocadas no preâmbulo e inseridas automaticamente no documento por meio de comandos como `\maketitle` ou `\titlepage`.

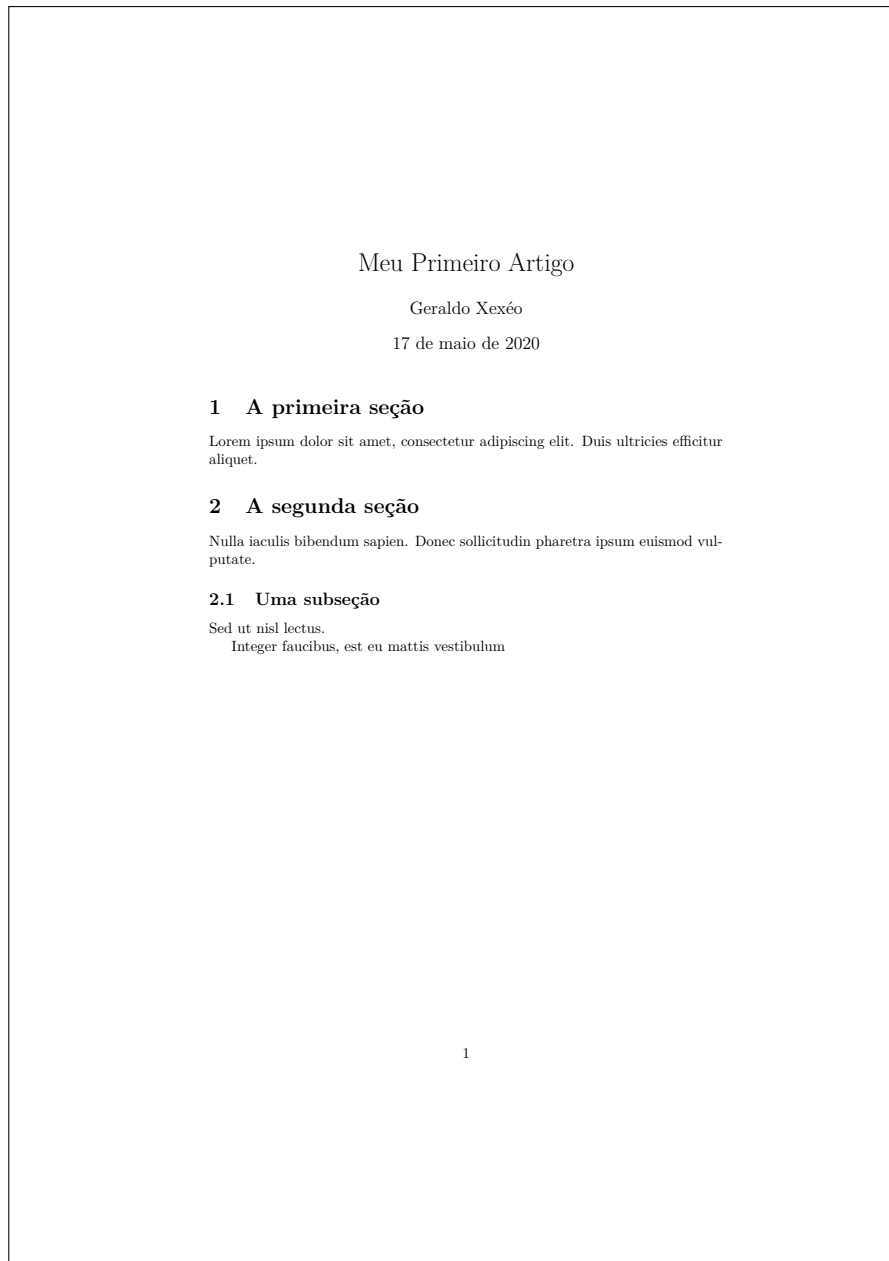


Figura 3.2: PDF produzido pelo texto da figura 3.2.

Um artigo pode possuir um título, um subtítulo, um ou mais autores e uma data. A data, se não for colocada, é gerada automaticamente. A listagem 3.2 mostra os comandos `title`, `author`. O efeito deles acontece quando o comando `\maketitle` é chamado. O resultado é mostrado na figura 3.2.

## 3.4 Comandos mais comuns

A figura 3.3 mostra alguns comandos mais comuns para formatar caracteres e seus resultados. Atenção ao uso das linhas vazias para manter ou trocar de parágrafo:

---

```

1 \textbf{negrato}
2 \textit{itálico}
3
4 \underline{sublinhado}
5
6 \Large
7 Texto Grande
8 \normalsize
9
10 e\textsuperscript{superescrito}
11 e\textsubscript{subescrito}

```

---

<b>negrato</b> <i>itálico</i> <u>sublinhado</u> Texto Grande e <sup>superescrito</sup> e <sub>subescrito</sub>
---

Figura 3.3: Comandos comuns em L<sup>A</sup>T<sub>E</sub>X

### 3.4.1 Fazendo referência a outra parte do documento

Muitas vezes em um documento é necessário citar alguma outra parte do mesmo, como uma figura, tabela ou seção. Para isso são usados dois comandos `\label{<rótulo>}`, que cria um rótulo para o local com o conhecimento do número a ser usado, como o número da figura, e `\ref{<rótulo>}`. Existe outras opções de citação específica como `\pageref{<rótulo>}`, que cita a página e não o contexto.

Listagem 3.3: Artigo usando as referências.

---

```
1 \documentclass{article}
2 \usepackage[T1]{fontenc}
3 \usepackage[english,brazilian]{babel}
4
5 \title{Artigo com rótulos, itemize e enumerate}
6
7 \author{Geraldo Xexéo}
8
9 \begin{document}
10
11 \maketitle
12
13 \section{A primeira seção}\label{sec:pri}
14 Leia a seção \ref{sec:seg}, na página \pageref{sec:seg}.
15 Mas você pode continuar em \textbf{negrito}
16 ou \textit{itálico}.
17
18 \section{A segunda seção}\label{sec:seg}
19 Leia a seção \ref{sec:pri}
20
21 Uma lista de itens:
22 \begin{itemize}
23   \item Um
24   \item dois
25   \item três
26 \end{itemize}
27
28 E uma enumeração:
29 \begin{enumerate}
30   \item Um
31   \item dois
32   \item três
33 \end{enumerate}
34
35 \end{document}
```

---

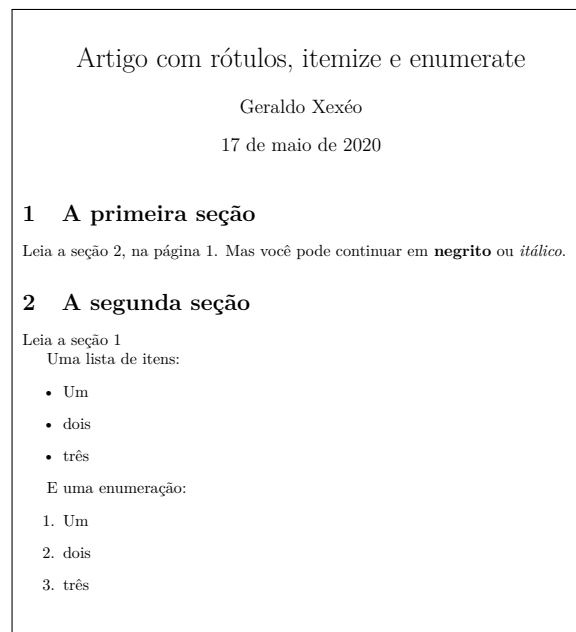


Figura 3.4: PDF do exemplo do uso de referências (listagem 3.3).

## 3.5 Usando pacotes

### 3.5.1 O Que São Pacotes

- São extensões que adicionam poder ao  $\text{\LaTeX}$
- Algumas são muito usadas
  - Babel – para escrever em outras línguas
    - ◊ Sim,  $\text{\LaTeX}$  é muito focado em inglês, mas há muitos pacotes criados em outras línguas, como alemão.
    - ◊ Um bom pacote é auto-configurável na maioria das línguas por meio do Babel
    - ◊ Um ótimo pacote deixa você configurar como você quiser a língua
- Comando
  - `\usepackage[<opções>]{package}`
- Os bons pacotes tem várias formas de configurações nessas opções
- Os bons pacotes estão vivos, isto é, são mantidos por seus autores ou seguidores

Três pacotes essenciais para escrever em português do Brasil são:

- Babel

- `inputencode`
- `fontencode`

### 3.5.2 Babel

- Pacote que permite usar o *L*<sup>A</sup>T<sub>E</sub>X com outras linguagens
- Chamado por pacotes que escrevem algo, como a palavra “Capítulo”
- Comando
  - `\usepackage[english,brazilian]{babel}`
  - A última linguagem é a mais importante
- A opção “portuguese” usa termos de Portugal, como dizer que um documento Web foi *acedido* em vez de *acessado*.
- Usar sempre **brazilian**

### 3.5.3 inputencode

- `\usepackage[utf8]{inputencode}`
- Faz o *L*<sup>A</sup>T<sub>E</sub>X entender código UTF-8 nos documentos que ele lê
  - Caracteres acentuados do Português e outras línguas!
    - ◊ áéíóúâêîôûäëïöüàèìòù
- Você não precisa mais usar `\te`
- **Não use o utf8x**, ele morreu e tem defeitos
- Melhor ainda, use o Lua*L*<sup>A</sup>T<sub>E</sub>X e dispense esse pacote!

### 3.5.4 fontencode

- `\usepackage[T1]{fontencode}`
- Quando gera o PDF, o *L*<sup>A</sup>T<sub>E</sub>X por *default* use o *font encoding* OT1, que só tem 7 bits.
  - Isso faz que uma leitura do texto do PDF para indexação, por exemplo, recupere combinações de caracteres que foram geradas para representar um caracter
    - ◊ Isso gera problemas na indexação do seu documento, ruim para você
- Garante também que as ligaduras, grande parte da beleza do texto do T<sub>E</sub>X sejam geradas, pois elas não são construídas, mas sim pertencentes as fontes.
- Sempre necessário

## 3.6 Ambientes

Ambientes são escopos fechados que são usados para mudar, temporariamente, o comportamento do  $\text{\LaTeX}$  basicamente criando um contexto onde algo pode ser feito de acordo com regras específicas, como a criação de listas de itens, equações, figuras, etc. Em geral, um comando dado dentro do ambiente deixa de ser válido fora do ambiente.

Uma ambiente é marcado com um início e um fim, usando os comandos `\begin{<nome do ambiente>}` e `\end{<nome do ambiente>}`. Ambientes podem ser construídos um dentro do outro, de forma estruturada.

### 3.6.1 Ambiente Mais Usados

- `itemize` – ver listagem 3.3
- `enumerate` – ver listagem 3.3
- `equation`
- `tabular` – usado normalmente dentro de um ambiente `table`
- `abstract`
- Floats – alguns ambiente “flutuam” no documento, sendo posicionados pelo algoritmo do  $\text{\LaTeX}$ .
  - `figure`
  - `table`
  - `lstlisting` – depende do pacote `listings`

### 3.6.2 Equações

Você pode fazer equações dentro do texto, o que é conhecido no  $\text{\LaTeX}$  como *inline*, ou em ambientes próprios, quando elas ficam isoladas, como mostrado na figura 3.5

Construir equações em  $\text{\LaTeX}$  é um aprendizado de uma sub-linguagem, que é, porém, bastante simples. Para iniciar esse aprendizado é interessante olhar sites que constroem equações interativamente, como <https://www.codecogs.com/latex/eqneditor.php>. Na verdade, basta buscar “*latex equation online*” no Google que encontrará rapidamente um site semelhante a imagem de figura 3.6.

### 3.6.3 Resumos

Um artigo pode ter um resumo, e a classe `article` provê um ambiente `abstract`.

## 3.7 Floats

### 3.7.1 O Que São Floats

- Ambientes que o *L*<sup>A</sup>T<sub>E</sub>X posiciona no melhor lugar possível de acordo com suas regras de diagramação.
  - `figure`
  - `table`
- Permite opções que orientam o posicionamento do bloco formatado gerado pelo algoritmo
  - `\begin{figure}[hbt]`
    - ◊ h – here – tenta colocar na posição onde o comando está em relação ao texto
    - ◊ b – bottom – tenta colocar no fim da página
    - ◊ t – top – tenta colocar no top da página
  - Essa é a melhor ordem, pois (quase) garante que a imagem não vai aparecer antes de ser citada
- Flutuar significa que você não determina exatamente onde vão ficar, mas sugere ao algoritmo

### 3.7.2 O Ambiente figure

Como o nome diz, o ambiente `figure` serve para inserir figuras em seu texto. Como a maior parte das pessoas faz figuras fora do *L*<sup>A</sup>T<sub>E</sub>X, mesmo havendo pacotes poderosíssimos de desenhos por comandos, ele é usado normalmente com o comando `\includegraphics[keyvals]{imagefile}`. Para isso é importante usar o pacote `graphicx`, que possui ainda comandos para fazer operações na figura, como cortar e colocar em escala, como fazemos com a opção `width` na 3.8.

### 3.7.3 Os Ambientes tabular e table

Para construir tabelas usamos o ambiente `tabular`, porém, para permitir que o *L*<sup>A</sup>T<sub>E</sub>X as coloque no lugar mais apropriado no texto, usamos o ambiente flutuante `table`

### 3.7.4 Ambientes para Listagens

Existem vários ambientes em *L*<sup>A</sup>T<sub>E</sub>X que permitem mostrar listagens de programas de computador e algoritmos. Para listagens de programas um bas-



tante poderoso e configurável, e que está sendo usado neste texto para mostrar os arquivos em  $\text{\LaTeX}$ , é o `lstlisting`, que faz parte do pacote `listings`<sup>1</sup>.

Para algoritmos, existem várias opções viáveis, como os pacotes `algorithms`, `alg`, sendo que o mais atualizado é o `algorithm2e`.

---

<sup>1</sup>Atenção ao “s”

---

```

1 Uma equação inline é construída a partir do caracter
2 \$, que no uso normal precisa ser feito com uma barra
3 antes, como em $e=\sum_{i=0}^n 1)/i!$.
4
5 Porém, para colocar a equação em destaque e poder citá-la
6 por uma referência numérica, como em equação \ref{eq:e},
7 deve ser usado o ambiente
8 \lstinline|equation|.
9
10 \begin{equation}\label{eq:e}
11 e=\sum_{i=0}^n 1)/i!
12 \end{equation}
13
14 Se o número da equação não for desejado, o ambiente
15 correto é o \lstinline|equation*|, porém para isso é
16 importante use o ótimo pacote
17 \lstinline|\usepackage{amsmath}|.
18
19
20 \begin{equation*}
21 e=\sum_{i=0}^n 1)/i!
22 \end{equation*}

```

---

Uma equação inline é construída a partir do caracter \$, que no uso normal precisa ser feito com uma barra antes, como em  $e = \sum_{i=0}^n 1)/i!$ . Porém, para colocar a equação em destaque e poder citá-la por uma referência numérica, como em equação 3.1, deve ser usado o ambiente `equation`.

$$e = \sum_{i=0}^n 1)/i! \quad (3.1)$$

Se o número da equação não for desejado, o ambiente correto é o `equation*`, porém para isso é importante use o ótimo pacote `\usepackage{amsmath}`.

$$e = \sum_{i=0}^n 1)/i!$$

Figura 3.5: Exemplo de uso de equações.

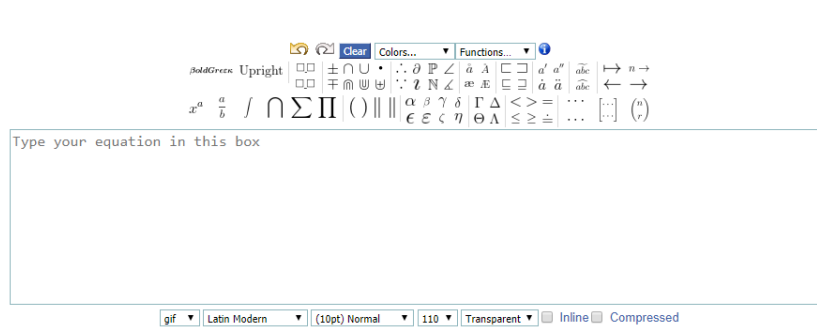


Figura 3.6: Exemplo de editor de equação on-line

---

```

1 \begin{figure}
2 \centering
3 \includegraphics[height=0.3\textheight]{Images/Picture6}
4 \caption{Capa do livro de \LaTeX\ }
5 \label{fig:picture6}
6 \end{figure}

```

---

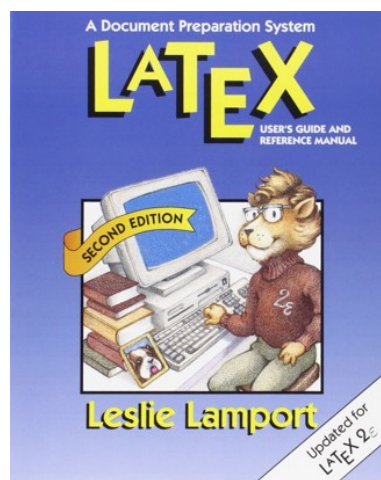
Figura 3.7: Capa do livro de  $\text{\LaTeX}$ 

Figura 3.8: Exemplo de uso do ambiente figure.

---

```

1  \begin{table}
2  \caption{Tabela de Idades}
3  \centering
4  \label{tab:idades}
5  \begin{tabular}{|c|c|}
6  \hline
7  \textbf{idade} & \textbf{nome} \\
8  \hline
9  0-2 & bebê \\
10  3-12 & criança \\
11  12-19 & adolescente \\
12  20-25 & jovem adulto \\
13  25-60 & adulto \\
14  60-80 & sênior \\
15  80- & terceira idade \\
16  \hline
17  \end{tabular}
18  \end{table}

```

---

Tabela 3.1: Tabela de Idades	
idade	nome
0-2	bebê
3-12	criança
12-19	adolescente
20-25	jovem adulto
25-60	adulto
60-80	sênior
80-	terceira idade

Figura 3.9: Exemplo de uso dos ambientes `table` e `tabular`.

## Capítulo 4

# Controle de Referências

BIBTEX é uma ferramenta e um formato de arquivos usados para descrever e processar listas de referência, normalmente em documentos L<sup>A</sup>T<sub>E</sub>X. Na prática, o termo é usado como termo geral para tratar de dois conjuntos distintos de aplicações associadas ao L<sup>A</sup>T<sub>E</sub>X:

- BIBTEX e biber, programas que processam a informação bibliográfica de um documento L<sup>A</sup>T<sub>E</sub>X e criam a bibliografia
- natbib e biblatex, pacotes L<sup>A</sup>T<sub>E</sub>X que formatam as citações e a bibliografia
  - natbib só funciona com BIBTEX
  - biblatex funciona com ambos processadores

### 4.1 O arquivo .bib

O funcionamento básico do BIBTEX depende de um arquivo .bib, exemplificado na figura 4.1, que é uma base de dados contendo referências contendo uma chave de citação. Essa base é construída em um arquivo texto comum, o que permite que seja manipulada tanto por programas específicos, quanto pelos autores diretamente. Ao longo do texto, o autor pode fazer citações a essas referências usando comandos como `\cite{Xexeo2020}`.

A interação entre o processador L<sup>A</sup>T<sub>E</sub>X, o pacote escolhido e o processador de referências escolhido, também conhecido com *backend engine*, trará para a versão a ser impressa as citações e a bibliografia no formato desejado.

Como pode ser visto na figura 4.1, um arquivo .bib é compostos de entradas, como `@book` que são, por sua vez, compostas de campos, obrigatórios ou opcionais, como `author`. As entradas geram formatos específicos na bibliografia, de acordo com as normas utilizadas.

```
@book{sommerville:requirements,  
author = {Sommerville, Ian and Sawyer, Pete},  
title = {Requirements Engineering: A Good Practice Guide},  
year = {1997},  
isbn = {0471974447},  
publisher = {John Wiley \& Sons, Inc.},  
address = {USA},  
edition = {1st}  
}  
  
@Article{therac25,  
author      = {Nancy G. Levenson and Clark S. Turner},  
title       = {An Investigation of the Therac-25 Accidentes},  
journaltitle = {Computer},  
date        = {1993-07},  
volume      = {26},  
number      = {7},  
pages       = {18-41},  
}
```

Figura 4.1: Exemplo de arquivo .bib

### 4.1.1 Tipos de Entradas Mais Usados

Os tipos de entradas mais usados são:

- article
- inbook
- book
- incollection
- collection
- inproceedings
- proceedings
- report
- thesis
- online
- misc

## 4.2 O Ecossistema BIBTEX

A figura 4.2(ebosi e user2478, 2015), descreve o ecossistema BIBTEX. Ele é formado por pacotes que executam no processador L<sup>A</sup>T<sub>E</sub>X, processadores para bibliografia, arquivos com bases de referência, programas que gerenciam esses arquivos (o que pode ser feito com editores de texto, no caso de arquivos .bib), e, possivelmente, outros programas que fazem transformações de arquivos. A figura mostra que há duas escolhas a serem feitas: o pacote L<sup>A</sup>T<sub>E</sub>X a ser usado e o processador a ser escolhido.



Figura 4.2: O Ecossistema BIBTEX(ebosi e user2478, 2015).

A figura 4.3<sup>1</sup> mostra o funcionamento básico quando o biblatex é usado. Um ciclo que usa o natbib pode precisar de mais uma compilação com o processador  $\text{\LaTeX}$  de escolha. Basicamente o processador  $\text{\LaTeX}$  gera um arquivo que contém as informações sobre as citações feitas no documento. Essa informação está no .bcf no caso do biblatex com o biber. Esse arquivo é usado, junto com os vários arquivos .bib contendo as bases de dados de referências utilizadas, para gerar um arquivo .bbl, que contém a bibliografia. Esse arquivo é usado, novamente pelo processador  $\text{\LaTeX}$ , para gerar o documento final.

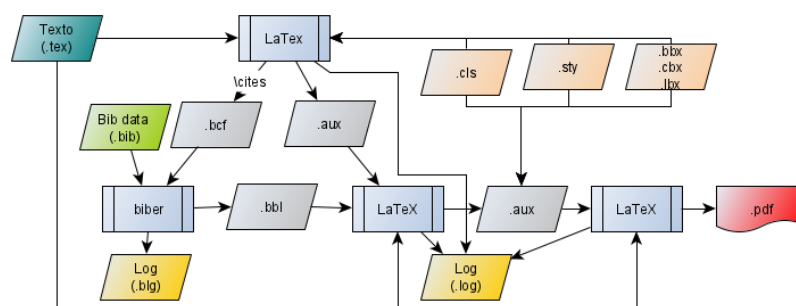


Figura 4.3: O fluxo de processamento (simplificado) do  $\text{\LaTeX}$

## 4.3 Comparando as Opções

Como se pode deduzir da figura 4.2, duas comparações básicas devem ser feitas: dos processadores, biber ou bibtex, e dos pacotes  $\text{\LaTeX}$ , natbib ou biblatex.

### 4.3.1 natbib vs biblatex

As duas principais opções de pacotes  $\text{\LaTeX}$  para referências são o natbib e o biblatex(Munn, 2015):

- **natbib** – é um pacote antigo e estável, porém ele é mantido mas não evoluído.
  - Vantagens
    - ◊ Vários formatos já definidos (arquivos .bst)
    - ◊ Possui um pacote custom-bib, com o aplicativo makebst, que gera estilos bibliográficos, .bst, iterativamente

<sup>1</sup>É uma cópia da figura 2.2.



- Desvantagens
  - ◊ Depende do `BIBTEX`, que tem algumas desvantagens.
  - ◊ `.bst` é difícil de fazer (linguagem posfixa)
  - ◊ Orientado para autor-ano e numérico, mas não autor-título, tipo de citação que aparece em outras áreas não tecnológicas
  - ◊ É fácil pegar a bibliografia gerada e incluir no arquivo `.tex`, o que algumas editoras pedem.
- **biblatex** – ativamente desenvolvido e em evolução, ligado ao *backend* mais poderoso biber. Substitui o `natbib`, e tem pacotes adicionais, como o **biblatex-abnt**.
  - Vantagens
    - ◊ Mais campos
    - ◊ Unicode no arquivo `.bib`, sem problemas para caracteres de outras línguas que podem aparecer em nomes de autores;
    - ◊ Usa métodos de `LATEX` para controle fino da sua bibliografia;
  - Desvantagens
    - ◊ Algumas revistas podem não aceitar (porque não fizeram o *upgrade* de seus estilos).
    - ◊ Não é fácil pegar a bibliografia criada e inserir no arquivo `.tex`, o que algumas editora pedem.

### 4.3.2 `BIBTEX` vs biber

Os dois principais processadores de referências são o `BIBTEX` e o biber:

- `BIBTEX`
  - Se for obrigado, use
  - Estável e debugado
  - Problemas com caracteres não padrão
  - Funciona com `natbib` e `biblatex`
- biber
  - Prefira usar o biber
  - Suporta UTF8!
  - ◊ `.bib` file muito mais verificado
    - ◊ Dá erros novos
  - ◊ Só funciona com `biblatex`

## 4.4 Usando biblatex e biber

Listagem 4.1: Exemplo de uso de biblatex

---

```

1 \usepackage{csquotes,xpatch}
2 \usepackage[backend=biber,style=numeric]{biblatex}
3 \addbibresource{references.bib}
4 ...
5 \cite{biber:2012}
6 ...
7 \printbibliography

```

---

Listagem 4.2: Controlando a forma de citação com o formato natbib.

---

```

1 \citet{biber:2012} não fala nada sobre isso.
2 Mas \citep{bibera2012} também não.
3 Por isso \citep{biber:2012} não tem erro.

```

---

#### 4.4.1 Exemplo dos comandos do biblatex

Listagem 4.3: Exemplo dos comandos do biblatex

---

```

1 \autocite{biber:2012}
2
3 \cite{biber:2012}
4
5 \parencite{biber:2012}
6
7 \textcite{biber:2012}
8
9 Lorem\footcite{biber:2012}
10
11 \fullcite{biber:2012}

```

---

### 4.5 Programas Externos

Vários são os programas externos que podem ser usados para gerenciar arquivos .bib, entre eles:

- JabRef (Java) – um gerenciador de referências que usa o arquivo .bib como seu formato padrão e pode ser integrado com IDEs L<sup>A</sup>T<sub>E</sub>X;
- Referencer (GNOME) – outro gerenciador de referências, só funciona no GNOME;

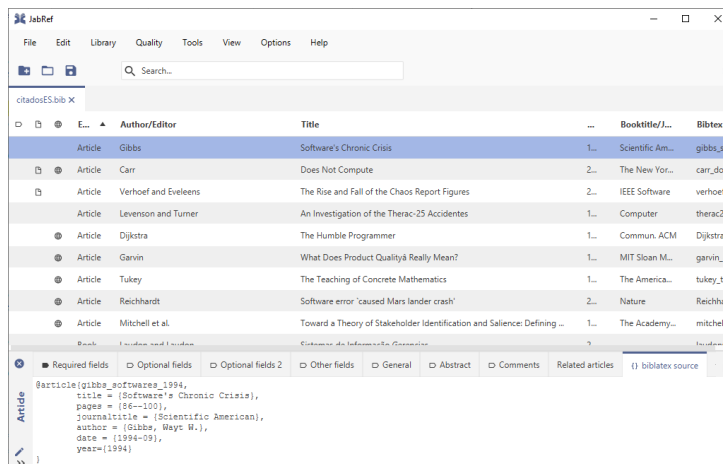


Figura 4.4: Exemplo de tela do Jabref.

- BiB2x – um processador que transforma do formato .bib para outros, e
- Zotero mais Better BibTEX for Zotero – uma extensão ao Zotero que permite gerar um arquivo .bib contendo as citações feitas em um documento .tex.

#### 4.5.1 JabRef

O JabRef, cuja a tela é apresentada na figura 4.4, é um programa com as seguintes características:

- Feito em Java
- Open Source
- Gerencia vários arquivos .bib em abas
- Permite unificar arquivos
- Suporte ao biblatex (UTF8,...)
- Detecta erros
- Busca na web



# Bibliografia

- Adams, Nico (mar. de 2007). *Polymer Informatics and The Semantic Web The Solution, Part 1: Adding Structure*. URL: <https://semanticsscience.wordpress.com/2007/03/30/polymer-informatics-and-the-semantic-web-the-solution-part-1-adding-structure/> (acesso em 03/2007).
- ebosi e user2478 (15 de ago. de 2015). *bibtex vs. biber and biblatex vs. natbib. answers to question*. T<sub>E</sub>X Stack Exchange. URL: <https://tex.stackexchange.com/questions/25701/bibtex-vs-biber-and-biblatex-vs-natbib/299286> (acesso em 12/05/2020).
- Lamport, Leslie (1994). *ΛT<sub>E</sub>X(2nd Ed.): A Document Preparation System: Users Guide and Reference Manual*. USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0201529831.
- Mittelbach, Frank et al. (1999). *The LaTeX Companion (Tools and Techniques for Computer Typesetting)*. 2<sup>a</sup> ed. USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0201362996.
- Munn, Alan (15 de ago. de 2015). *bibtex vs. biber and biblatex vs. natbib. Some terminology*. T<sub>E</sub>X Stack Exchange. URL: <https://tex.stackexchange.com/questions/25701/bibtex-vs-biber-and-biblatex-vs-natbib/299286> (acesso em 12/05/2020).
- Reid, Brian K. (out. de 1980). “Scribe: A Document Specification Language and its Compiler”. Tese de dout. Department of Computer Science, Carnegie-Mellon University.